CS 583 – Computational Audio -- Fall, 2021

Wayne Snyder
Computer Science Department
Boston University

Lecture 4. Amplitude Modulation continued…

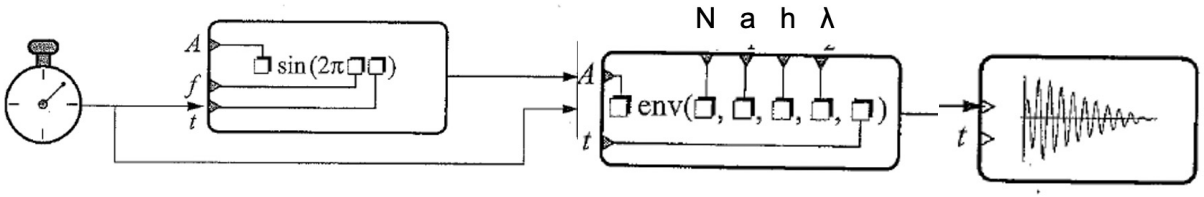
- Amplitude envelopes for individual notes
- Tremolo
- Ring Modulation
- If time: Vibrato and Frequency modulation

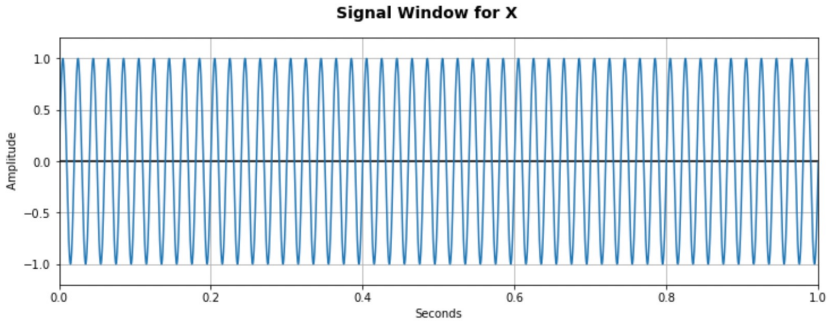
Computer Science

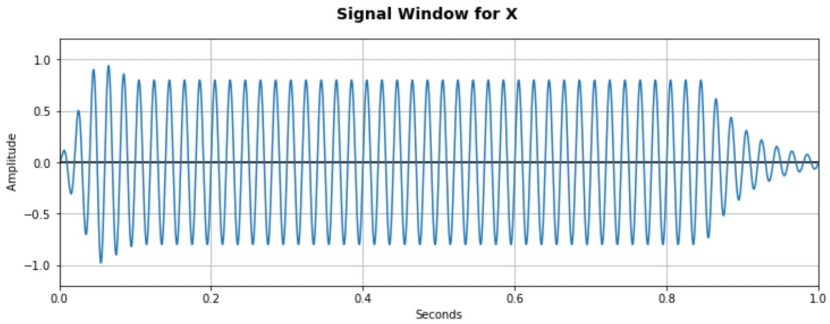
Digital Audio: Amplitude Envelopes

Modular Organization of Synthesis Program

A very typical filter module is one that shapes the amplitude envelope of the signal:

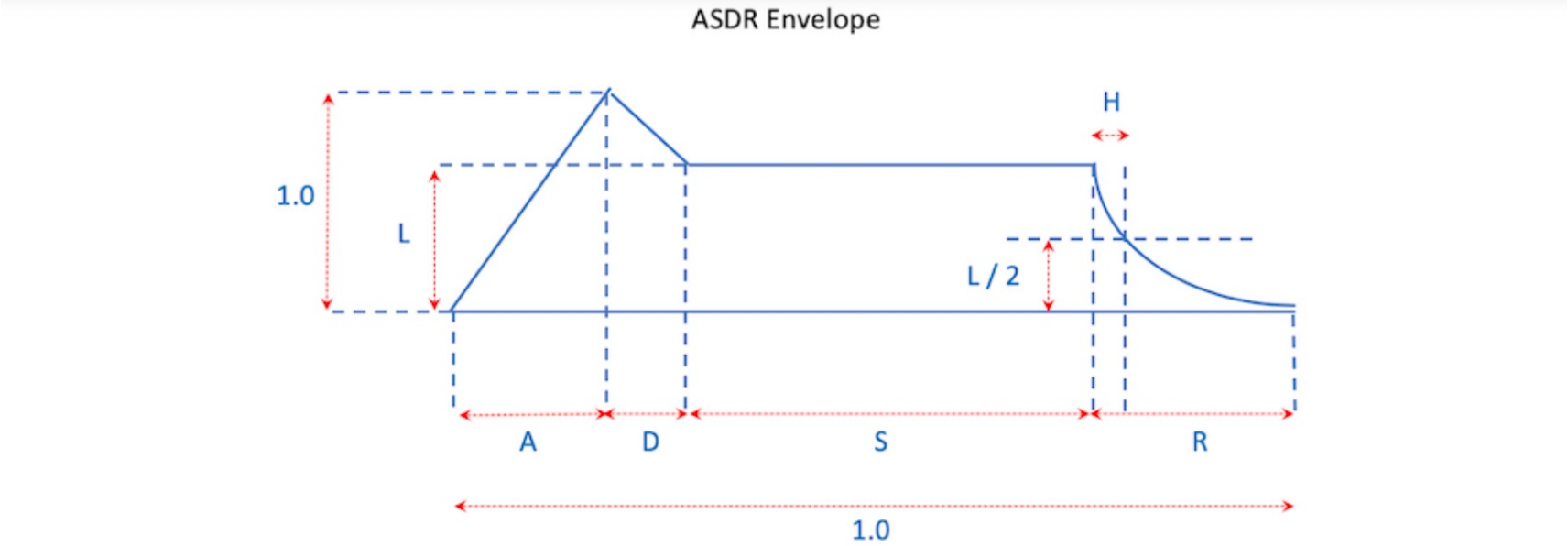






Digital Audio: Amplitude Envelopes

ASDR Envelopes are a common way to specify amplitude envelopes:

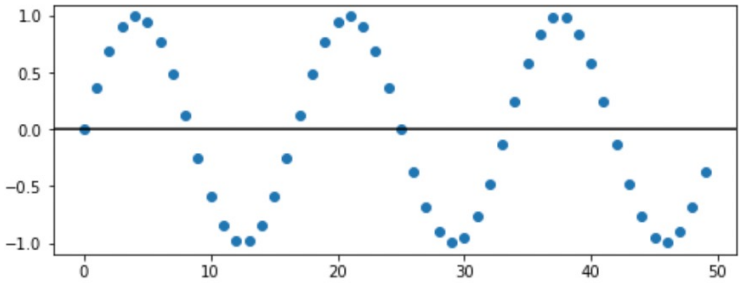


- A, D, S, R = relative length of each phase as a percentage of the whole envelope;
- L = relative level during sustain phase, as percentage of maximum, e.g., 0.5 is sustain at half max value; and
- H = relative half-life of exponential decay during R phase (time for amplitude to be reduced by half), as a percentage of the whole envelope.

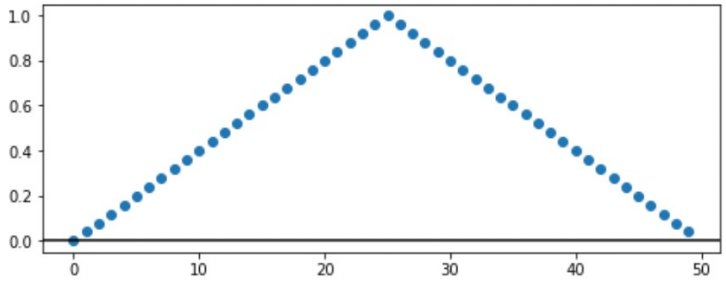
Note carefully that all the units are relative values in the range $[0..1]$ and that we must have $A + D + S + R = 1.0$.

Digital Audio: Amplitude Envelopes

In Python this is very simple to implement: We create a “signal” consisting of the scaling factors at each sample, and then just multiply the two signals; here is an example of a “swell” envelope:

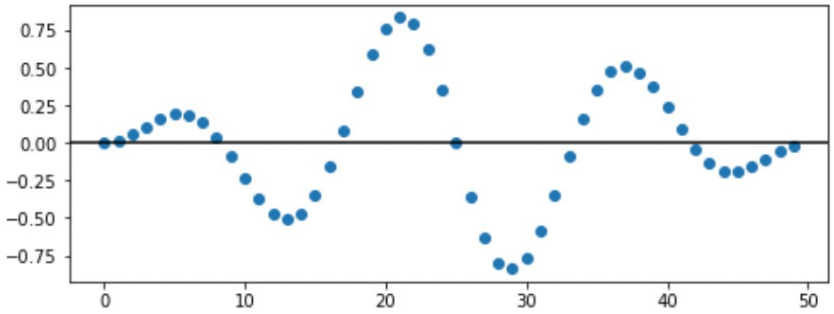


X



Y

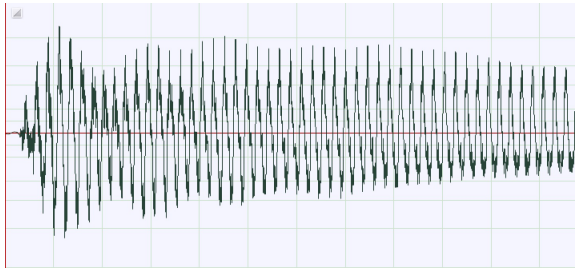
*

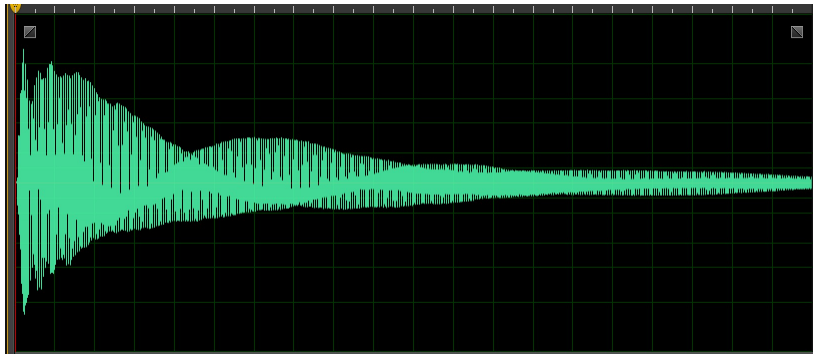


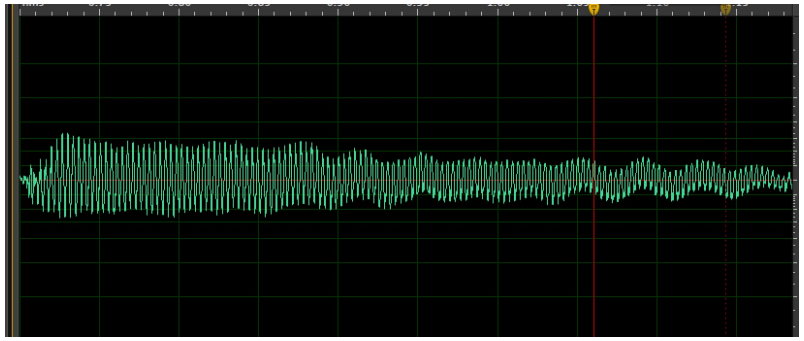
X * Y

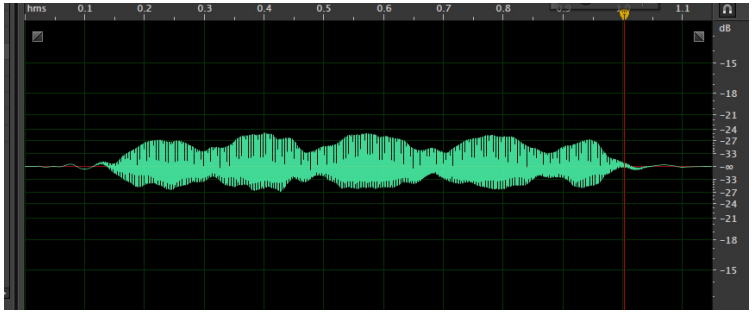
Amplitude Modulation

Of course, this is only a very simplified model of the reality of music signals:





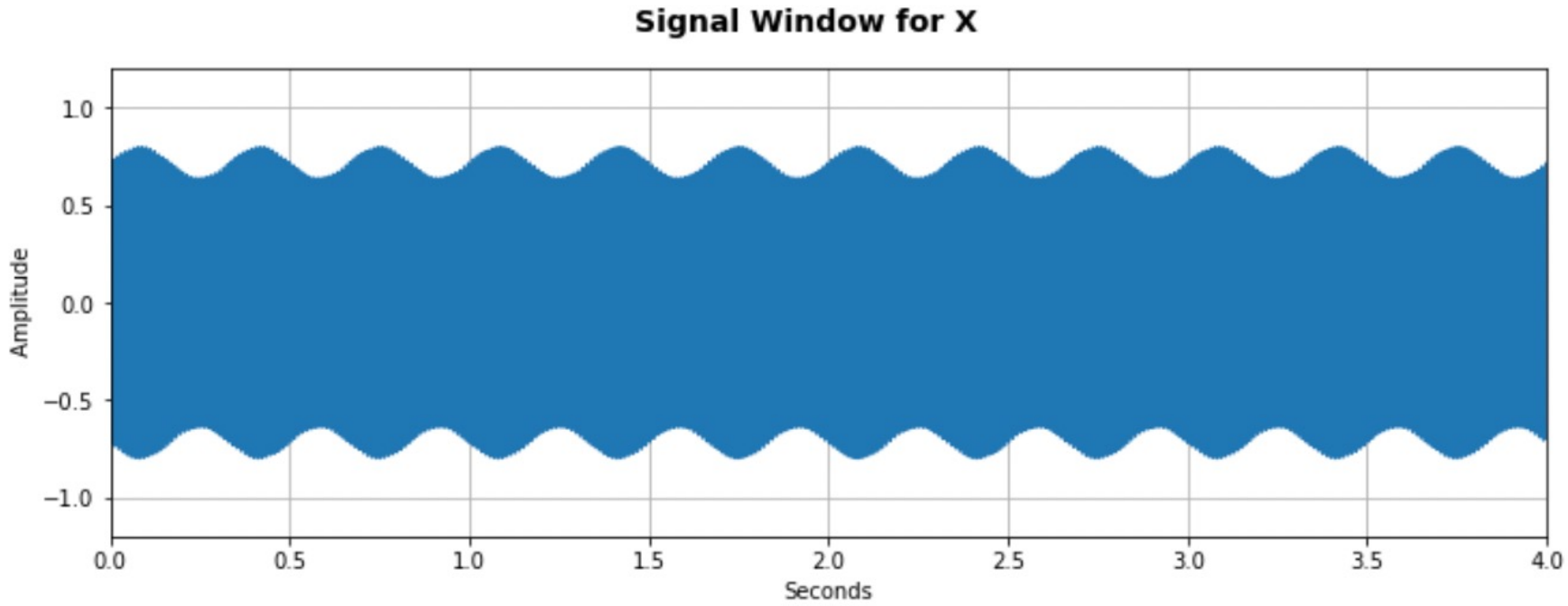




One common attribute of amplitude in musical signals is a regular variation in the amplitude, called **Tremolo**….

Amplitude Modulation

Tremolo is another kind of amplitude envelope, created by varying the amplitude according to a (relatively) slow sine wave:



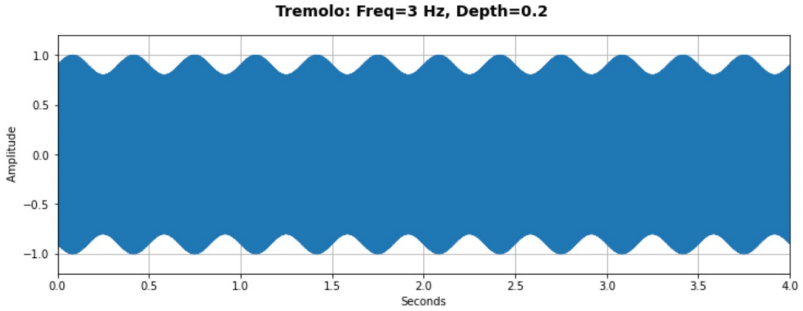
Amplitude Modulation: Tremolo

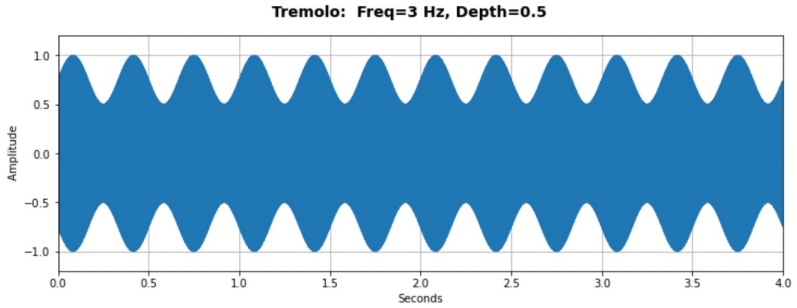
Tremolo is a common feature in electronic organs, the best example being the Hammond B3, beloved of R & B and Rock groups; the Lesley speakers in the B3 rotate to create its characteristic sound:

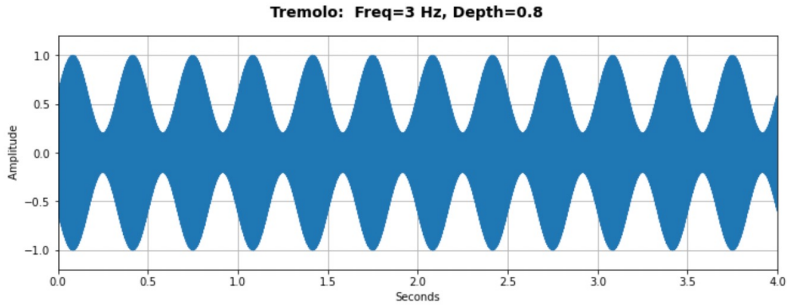
Amplitude Modulation: Tremolo

Computer Science

But what happens when you play around with the speed and amplitude of the tremolo?



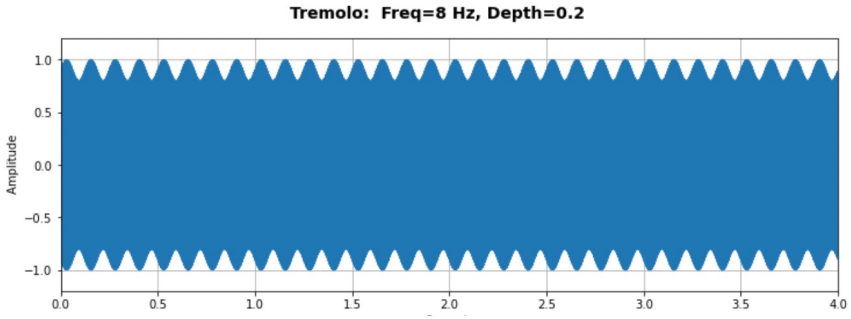


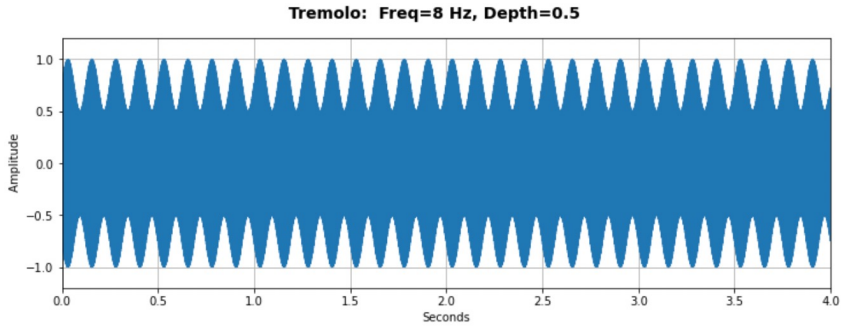


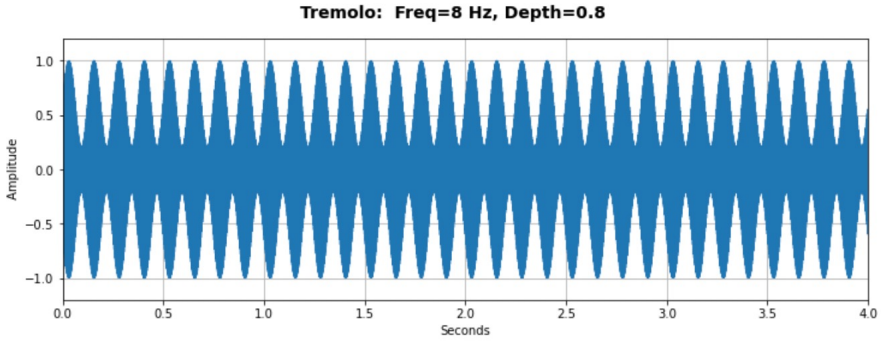
Amplitude Modulation: Tremolo

Computer Science

But what happens when you play around with the speed and amplitude of the tremolo?



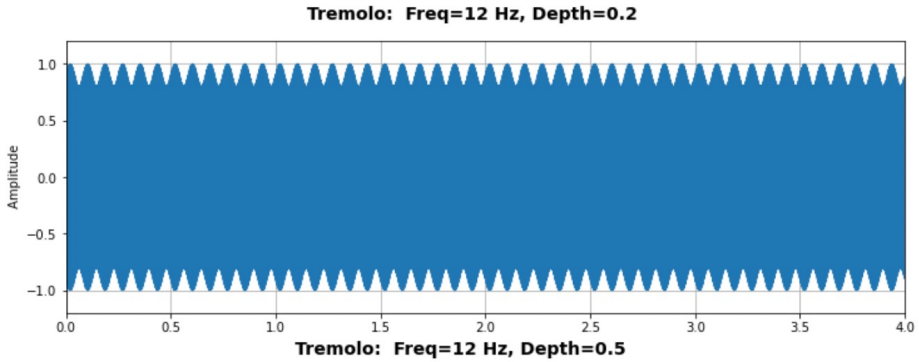


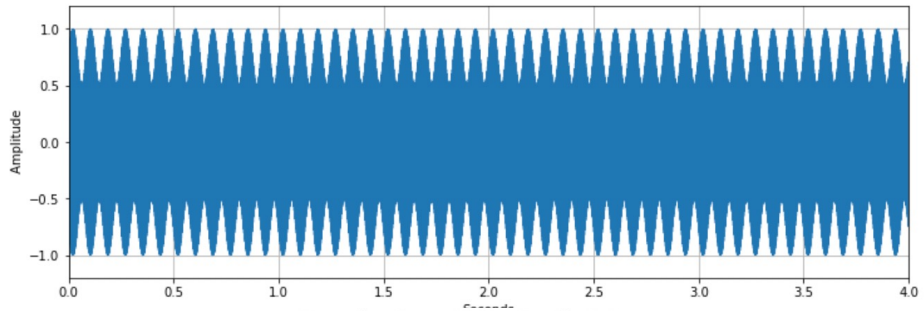


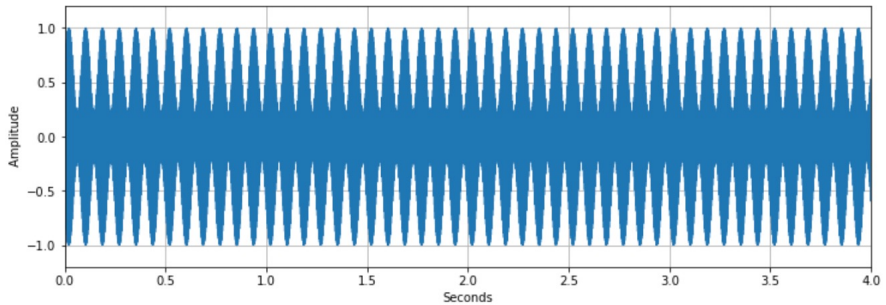
Amplitude Modulation: Tremolo

Computer Science

But what happens when you play around with the speed and amplitude of the tremolo?

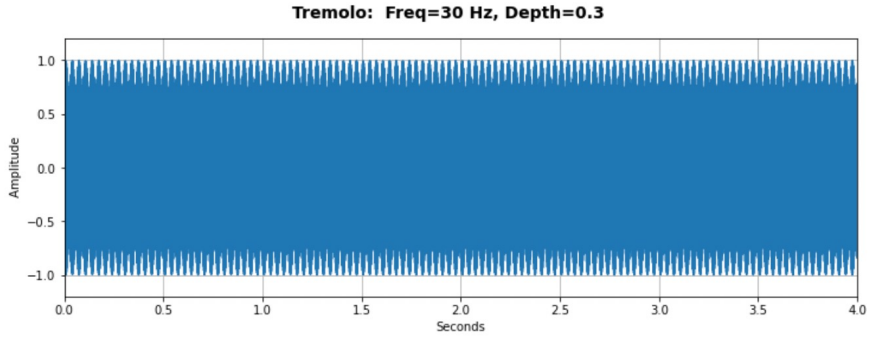


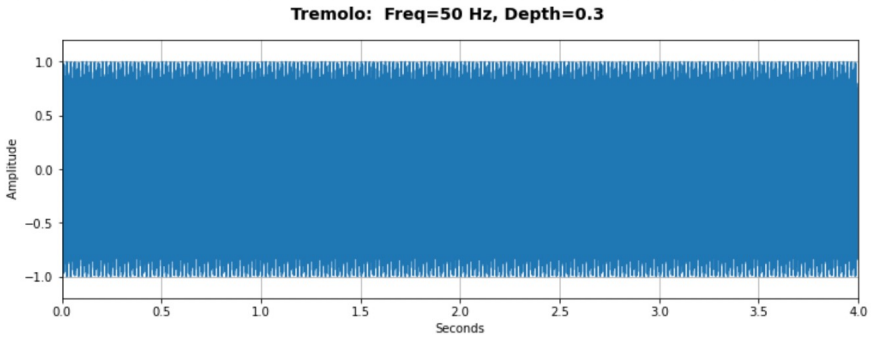


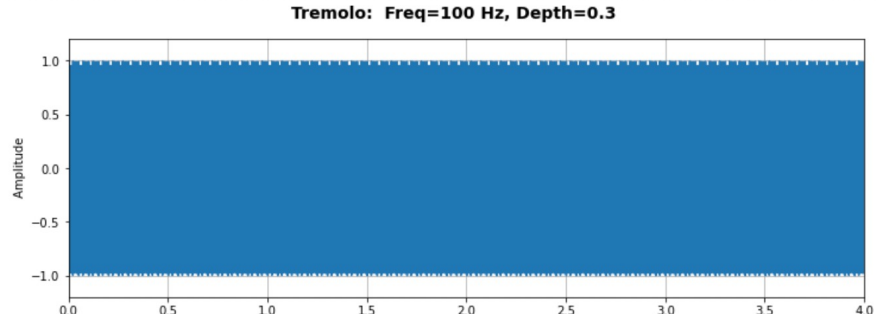


Amplitude Modulation: Tremolo

When the frequency of the tremolo is so fast you can not hear the individual “pulses,” it becomes part of the timbre of the sound:







Digital Audio: Amplitude and Ring Modulation

Tremolo allows the amplitude to be varied in the range [0 … 1.0], however, this is somewhat arbitrary.

Ring Modulation allows the amplitude to be varied as a full sine wave, in the range [-1.0 … 1.0], usually as part of a waveform generated by a Modulating Oscillator (whose output waveform is connected to the *amplitude* input of a Carrier Oscillator):

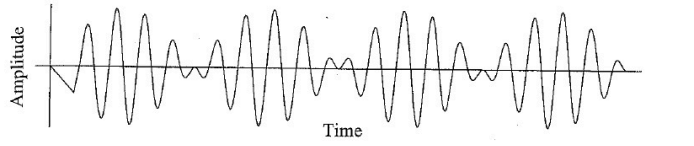


Figure 9.20
Amplitude modulation, carrier present in the output.

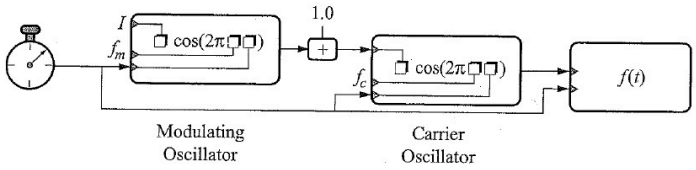
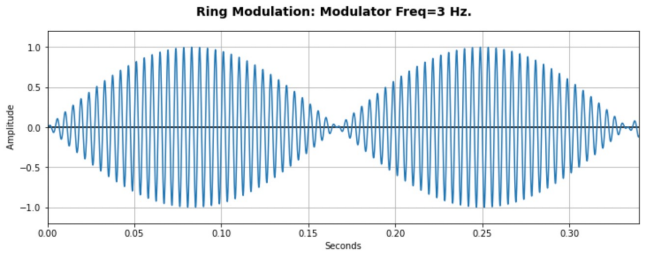


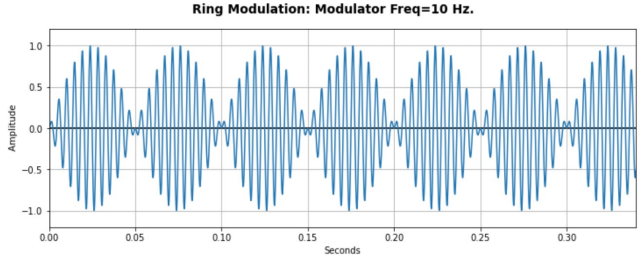
Figure 9.21
Amplitude modulation.

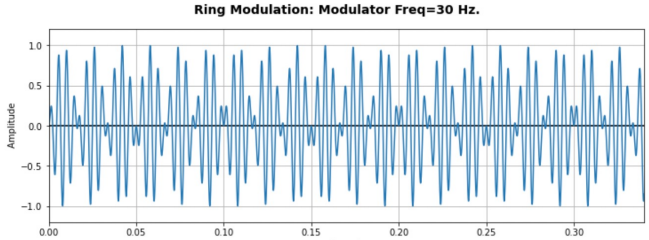
```
C = np.array( [  np.sin(2 * np.pi * carrier_freq * k / 44100) for k in range(44100) ] )
M = np.array( [  np.sin(2 * np.pi * modulation_freq * k / 44100) for k in range(44100) ] )

X = C * M
```

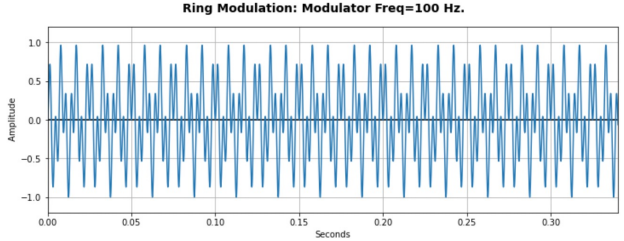
Amplitude Modulation: Ring Modulation

Again, let us consider the effect of various ring modulating frequencies with a simple sine wave of frequency 220 Hz:









Amplitude Modulation: Ring Modulation

Computer Science

But you can apply ring modulation to ANY carrier signal!

Audio samples of the ring modulation effect:

Unprocessed original sample

0:00 CC MENU

Ring modulation with a 2500 Hz sound

0:00 CC MENU

Notice the bell-like sound.

Ring modulation with an exponential sweep from 0 Hz to 9 kHz

0:00 CC MENU

On lower modulation frequencies, the ring modulation is perceived as a tremolo effect (as in the first part of the sound)

Dr. Who Dalek Voice:

My Voice:

My Dalek Voice:

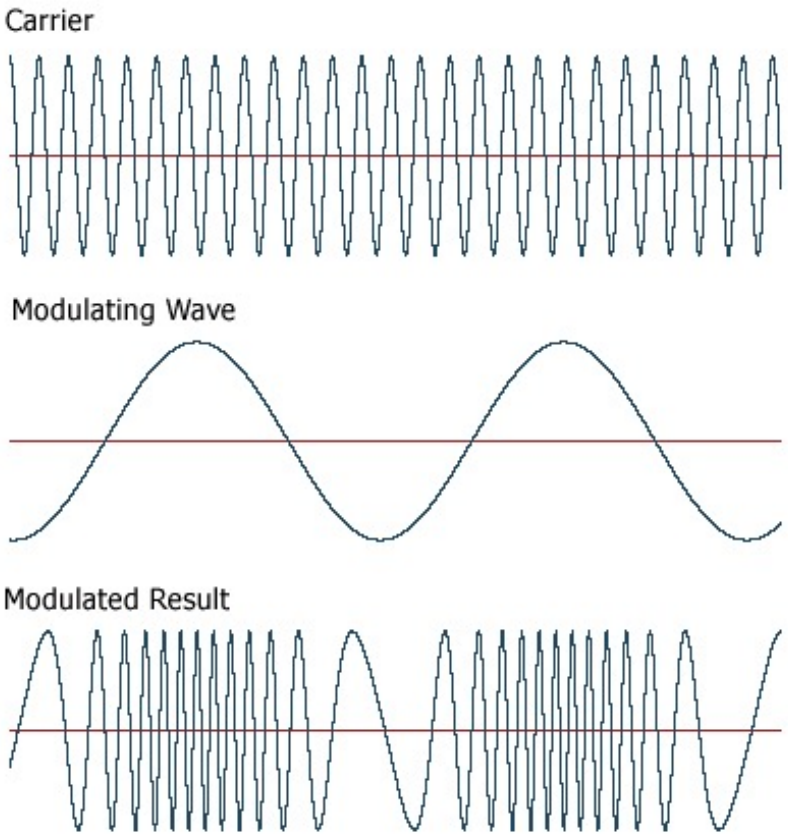
Clarinet with 30 Hz mod:

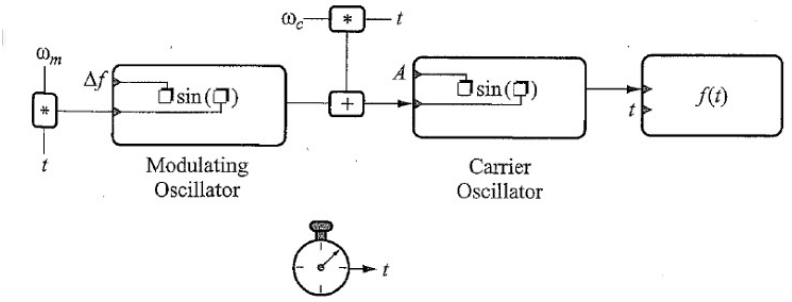
Clarinet with 1000 Hz mod:

Sound track from Forbidden Planet:

Digital Audio: Frequency Modulation

Frequency Modulation makes one simple but significant change to Ring Modulation: send the output of the Modulating Oscillator to the *frequency* input of the Carrier Oscillator:

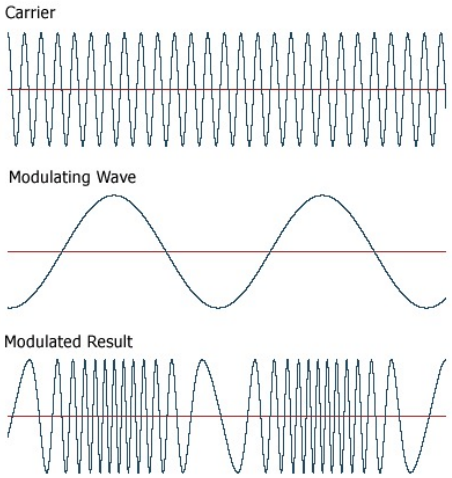




At slower modulating frequencies, this is called Vibrato, and is a standard part of instrumental and vocal technique (except for piano and percussion instruments!).

Digital Audio: Frequency Modulation

At slower modulating frequencies, this is called Vibrato, and is a standard part of instrumental and vocal technique (except for piano and percussion instruments!).



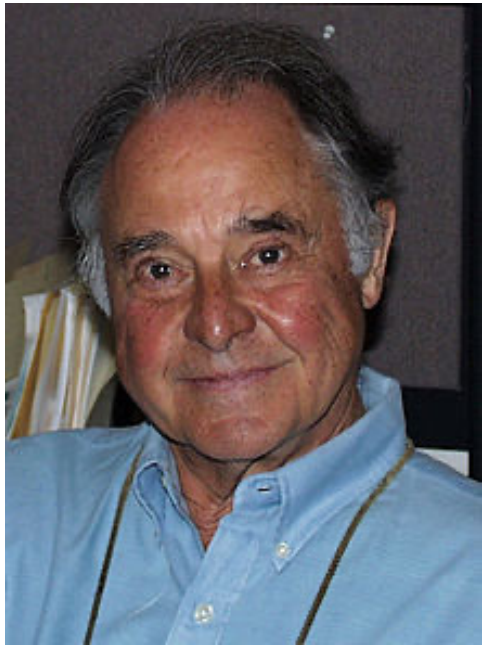
Classical singing technique specifies that the "ideal" vibrato is about 6 Hz with an amplitude of 0.25 to 0.5 semitones:

Violin vibrato:

Saxophone vibrato:

Digital Audio: Frequency Modulation

Frequency Modulation was invented by composer John Chowning in 1967. He is currently on the faculty at MIT. The outlines of the history of this technique are nicely summarized on the Wiki page for Chowning:



"Chowning's breakthrough allowed for simple yet rich sounding timbres, which synthesized 'metal striking' or 'bell like' sounds, and which seemed incredibly similar to real percussion. (Chowning was also a skilled percussionist). He spent six years turning his breakthrough into a system of musical importance and eventually was able to simulate a large number of musical sounds, including the singing voice. In 1973 Stanford University licensed the discovery to Yamaha in Japan, with whom Chowning worked in developing a family of synthesizers and electronic organs. This patent was Stanford's most lucrative patent at one time, eclipsing many in electronics, computer science, and biotechnology.

The first product to incorporate the FM algorithm was Yamaha's GS1, a digital synthesizer that first shipped in 1981. Some thought it too expensive at the time, Chowning included. Soon after, in 1983, Yamaha made their first commercially successful digital FM synthesizer, the DX7."

Digital Audio: Frequency Modulation

Frequency Modulation is the basis for a great deal of the "synthesizer" sound which you have probably been hearing your whole life, especially if you watched old space operas:

Ray Gun Blast: 

But it also can be used to create realistic instrument sounds such as bells:

Here is a nice composition based on FM:

http://www.youtube.com/watch?v=RIYEmwg_EyY

Frequency Modulation

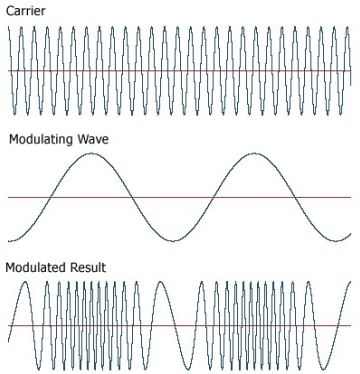
Here are some examples of Frequency Modulation with various frequencies and amplitudes, all modifying a straight A 440 Hz signal:

Amplitude (Hz)	Frequency (Hz)			
10	2	100	200	
10	6			
50	6			
10	10			
10	20			
20	50			
20	100			
20	200			

Frequency Modulation

There is a little problem with a straight-forward implementation of frequency modulation; here is a naïve way of changing the frequency of the audio signal:

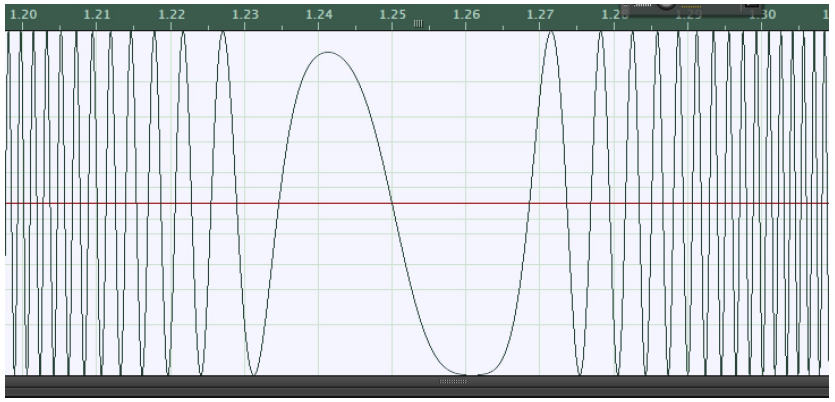
```
# Take a signal of freq f1 and amplitude A1 and modify it using freq modulation
# of freq f2 and amplitude A2

def freqModulationNaive(f1,A1,f2,A2,duration):
    X = [0] * (SR * duration)
    for k in range( SR*duration ):
        freqIncr = A2 * np.sin(2*np.pi*f2*k/SR)
        X[k] = A1*MAX_AMP*np.sin(2*np.pi*(f1+freqIncr)*k/SR)
    return X
```



So you would expect that

X = freqModulationNaive(440,1.0,6,10,5)

Would vary the 440 audio frequency by 10 Hz at the rate of 6 Hz, oscillating between 430 Hz and 450 Hz 6 times a second. But here is what you get:



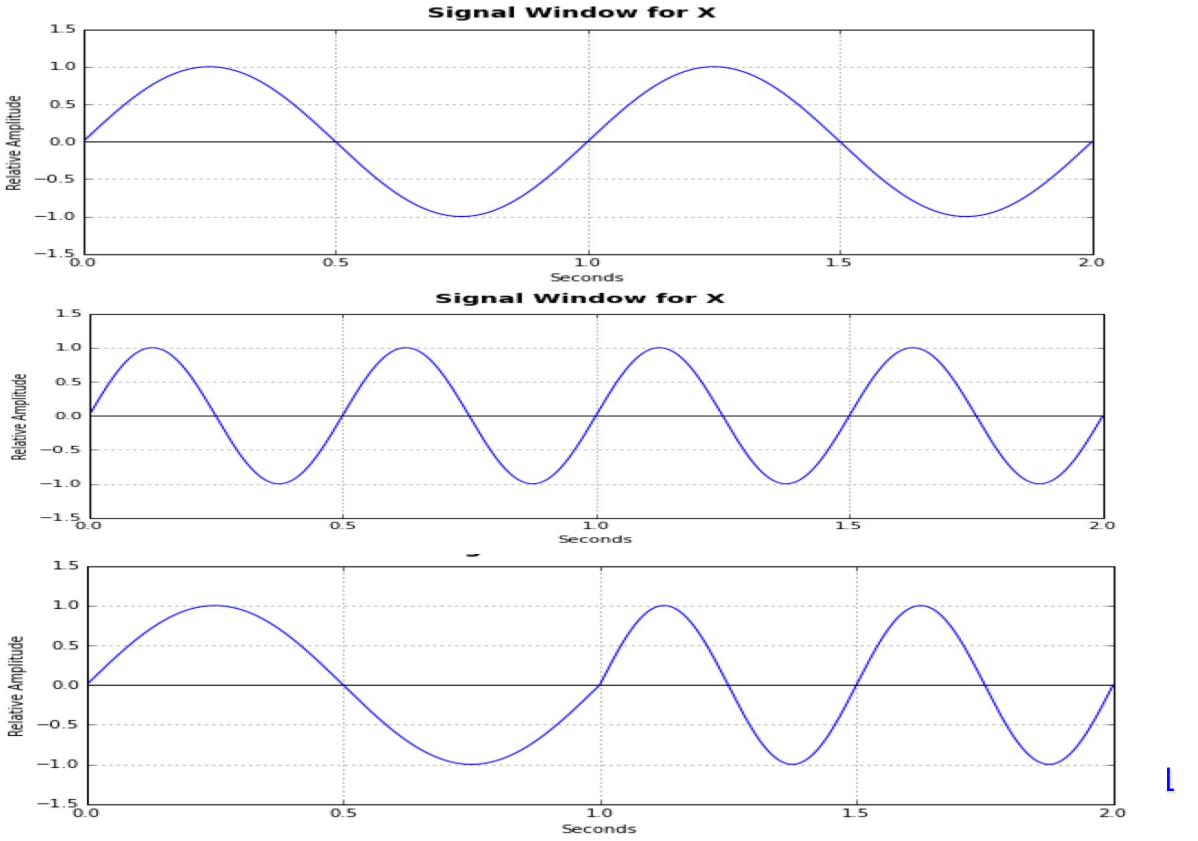
Frequency Modulation

What is the problem? When you change the frequency, but only keep track of time since the beginning of the signal, the (instantaneous) phases may not match.

Let's consider what happens when you change a signal suddenly between two frequencies, say 1 Hz and 2 Hz. If we have a 2 sec signal, and change at the 1 sec mark, we get a (somewhat) smooth transition, because the (instantaneous) phase at the transition point was the same:

$\sin(2 * \pi * 1 * t)$

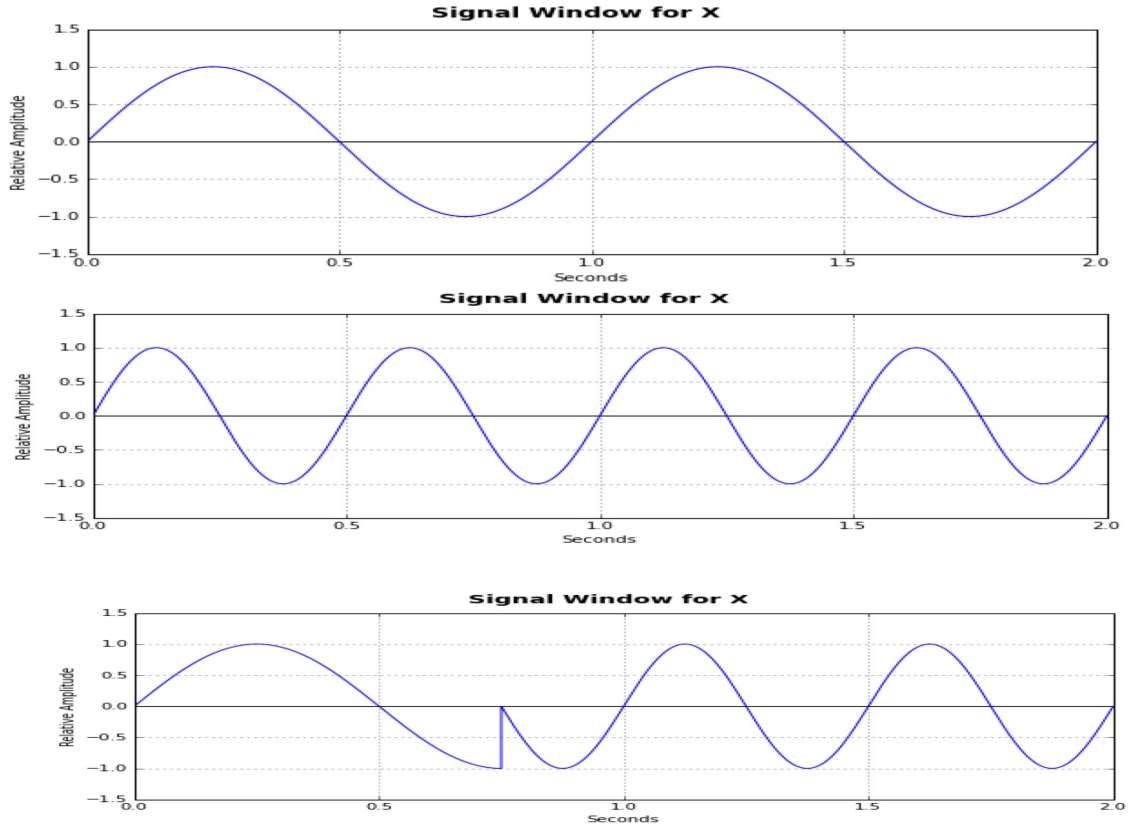
$\sin(2 * \pi * 2 * t)$



Frequency Modulation

What is the problem? When you change the frequency, but only keep track of time since the beginning of the signal, the (instantaneous) phases may not match.

If we change the frequency at 0.75 sec, however, the (instantaneous) phases do not match, and we get a discontinuity in the signal:



Frequency Modulation

Computer Science

Here is the corrected function:

```
def freqModulation(f1,A1,f2,A2,duration):
    X = [0] * (SR * duration)
    phase = 0.0
    newFreq = f1
    for k in range( SR*duration ):
        freqIncr = A2 * np.sin(2*np.pi*f2*k/SR)  # modulating signal
        oldFreq = newFreq
        newFreq = f1 + freqIncr
        phase +=  2 * pi * (k / SR) * (oldFreq - newFreq)
        X[k] = A1*MAX_AMP*np.sin(2*np.pi*newFreq*k/SR + phase)
    return X
```

Note that the phase has to be updated every time through the loop, to keep a running count of how far the phase has shifted each time the frequency is changed!